Practitioner's Docket No. 3193/102 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Vladimir Miloushev, Peter Nickolov

Application No.: 10/043,413
Filed: 01/10/2002
For: File Switch and Switched File System

Group No.: 2142
Examiner: Donaghue, Larry D.

Mail Stop Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PETITION TO WITHDRAW HOLDING OF ABANDONMENT BASED ON EVIDENCE THAT A REPLY WAS TIMELY MAILED OR FILED

1. I hereby petition to withdraw the holding of abandonment in this case, on the basis that a reply to the Notice of Allowance of 8/16/07 was timely filed.

2. I hereby state:

(a) A Request for Continued Examination, including a supplemental Information Disclosure Statement, was filed on October 30, 2007 by first class mail.

(b) The date-stamped return postcard from the PTO was not received.

(c) As of today's date, there is no indication on PAIR that the RCE was received by the PTO.

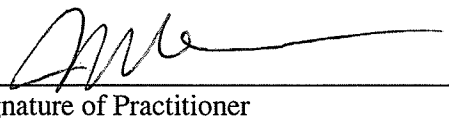
3. Attached is a copy of the RCE and IDS as filed on October 30, 2007. Please note that our file copy of the submission includes only the first page of each submitted reference. Complete copies of the references will be submitted to the PTO under separate cover.

4. As the person who signed the certificate of mailing, I hereby attest to personally placing the RCE with supplemental IDS and references in an envelope (box) addressed to Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 and handing the envelope to our firm's mail clerk, Colin Hoyle, who, as I recall, remained at the office after normal business hours specifically for the purpose of mailing the envelope, and, to the best of my knowledge and belief, did in fact hand-deliver the envelope to the U.S. Post Office at Boston's South Station on October 30, 2007 with sufficient postage as first class mail.

5. In consideration of these submissions, it is respectfully requested that the holding of abandonment be withdrawn.

6. I believe that no fee is required for this petition. However, please charge Deposit Account 19-4972 for any fees that are required by this paper.

Date: December 21, 2007



Signature of Practitioner
Jeffrey T. Klayman
Reg. No. 39,250
Bromberg & Sunstein LLP
125 Summer Street
Boston, MA 02110

03193/00102 790645.1

PTO/SB/30 (10-07)
Approved for use through 10/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Request for Continued Examination (RCE) Transmittal

Address to:
Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	10/043,413
Filing Date	January 10, 2002
First Named Inventor	Vladimir Miloushev
Art Unit	2142
Examiner Name	Prieto, Beatriz
Attorney Docket Number	3193/102

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. Submission required under 37 CFR 1.114 Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

   a. ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.

      i. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____

      ii. ☐ Other _____

   b. ☒ Enclosed

      i. ☐ Amendment/Reply

      ii. ☐ Affidavit(s)/ Declaration(s)

      iii. ☒ Information Disclosure Statement (IDS)

      iv. ☐ Other _____

2. Miscellaneous

   a. ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

   b. ☐ Other _____

3. Fees The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

   a. ☒ The Director is hereby authorized to charge the following fees, any underpayment of fees, or credit any overpayments, to Deposit Account No. 19-4972. I have enclosed a duplicate copy of this sheet.

      i. ☒ RCE fee required under 37 CFR 1.17(e)

      ii. ☐ Extension of time fee (37 CFR 1.136 and 1.17)

      iii. ☐ Other _____

   b. ☐ Check in the amount of $ _____ enclosed

   c. ☐ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Signature		Date	October 30, 2007
Name (Print/Type)	Jeffrey T. Klayman	Registration No.	39,250

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop RCE, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 or facsimile transmitted to the U.S. Patent and Trademark Office on the date shown below.

Signature			
Name (Print/Type)	Jeffrey T. Klayman	Date	October 30, 2007

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

DUPLICATE

PTO/SB/30 (10-07)
Approved for use through 10/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Request for Continued Examination (RCE) Transmittal

Address to:
Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Application Number	10/043,413
Filing Date	January 10, 2002
First Named Inventor	Vladimir Miloushev
Art Unit	2142
Examiner Name	Prieto, Beatriz
Attorney Docket Number	3193/102

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8, 1995, or to any design application. See Instruction Sheet for RCEs (not to be submitted to the USPTO) on page 2.

1. Submission required under 37 CFR 1.114 Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the order in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s) entered, applicant must request non-entry of such amendment(s).

   a. ☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.

      i. ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____

      ii. ☐ Other _____

   b. ☒ Enclosed

      i. ☐ Amendment/Reply

      ii. ☐ Affidavit(s)/ Declaration(s)

      iii. ☒ Information Disclosure Statement (IDS)

      iv. ☐ Other _____

2. Miscellaneous

   a. ☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of _____ months. (Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

   b. ☐ Other _____

3. Fees The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.

   a. ☒ The Director is hereby authorized to charge the following fees, any underpayment of fees, or credit any overpayments, to Deposit Account No. 19-4972. I have enclosed a duplicate copy of this sheet.

      i. ☒ RCE fee required under 37 CFR 1.17(e)

      ii. ☐ Extension of time fee (37 CFR 1.136 and 1.17)

      iii. ☐ Other _____

   b. ☐ Check in the amount of $ _____ enclosed

   c. ☐ Payment by credit card (Form PTO-2038 enclosed)

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

Signature		Date	October 30, 2007
Name (Print/Type)	Jeffrey T. Klayman	Registration No.	39,250

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop RCE, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 or facsimile transmitted to the U.S. Patent and Trademark Office on the date shown below.

Signature			
Name (Print/Type)	Jeffrey T. Klayman	Date	October 30, 2007

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Practitioner's Docket No. 3193/102 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Washington, D.C. 20231

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

NOTE: "An information disclosure statement shall be considered by the Office if filed by the applicant:
(1) Within three months of the filing date of a national application;
(2) Within three months of the date of entry of the national stage as set forth in section 1.491 in an international application; or
(3) Before the mailing date of a first Office action on the merits, whichever event occurs last." 37 C.F.R. section

---

CERTIFICATION UNDER 37 C.F.R. SECTIONS 1.8(a) and 1.10*
*(When using Express Mail, the Express Mail label number is **mandatory**;*
Express Mail certification is optional.)

I hereby certify that, on the date shown below, this correspondence is being:

MAILING

[X] deposited with the United States Postal Service in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

37 C.F.R. SECTION 1.8(a)

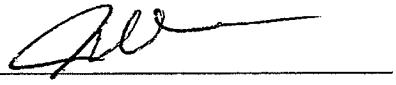
[X] with sufficient postage as first class mail.

37 C.F.R.SECTION 1.10*

[ ] as "Express Mail Post Office to Addressee"
Mailing Label No. _____ **(mandatory)**

TRANSMISSION

[ ]transmitted by facsimile to the Patent and Trademark Office.



Signature

Date: October 30, 2007

Jeffrey T. Klayman
(type or print name of person certifying)

WARNING:* *Each paper or fee filed by "Express Mail" **must have the number of the "Express Mail" mailing label placed thereon prior to mailing. 37 C.F.R. section 1.10(b).*
"Since the filing of correspondence under section 1.10 without the Express Mail mailing label thereon is an oversight that can be avoided by the exercise of reasonable care, requests for waiver of this requirement will not be granted on petition." Notice of Oct. 24, 1996, 60 Fed. Reg. 56,439, at 56,442.

1.97(b).

NOTE: "Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section." 37 C.F.R. section 1.56(a).
"Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:
(1) each inventor named in the application;
(2) each attorney or agent who prepares or prosecutes the application; and
(3) every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application." 37 C.F.R. section 1.56(c).

NOTE: The "duty as described in section 1.56 will be met so long as the information in question was cited by the Office or submitted to the Office in the manner prescribed by sections 1.97(b) - (d) and 1.98 before issuance of the patent." Notice of January 9, 1992, 1135 O.G. 13-25 at 17.

WARNING: "No information disclosure statement may be filed in a provisional application." 37 C.F.R. section 1.51(b).

List of Sections Forming Part of This Information Disclosure Statement

The following sections are being submitted for this Information Disclosure Statement:

(check sections forming a part of this statement: discard unused sections and number pages consecutively)

1. [x]Preliminary Statements
2. [x]Forms PTO/SB/08A and 08B (substitute for Form PTO-1449)
3. [x]Statement as to Information Not Found in Patents or Publications
4. [ ]Identification of Prior Application in Which Listed Information Was Already Cited and for Which No Copies Are Submitted or Need Be Submitted
5. [ ]Cumulative Patents or Publications
6. [x]Copies of Listed Information Items Accompanying This Statement
7. [ ]Concise Explanation of Non-English Language Listed Information Items
   - 7A. [ ]EPO Search Report
   - 7B. [ ]English Language Version of EPO Search Report
8. [ ]Translation(s) of Non-English Language Documents
9. [x]Concise Explanation of English Language Listed Information Items (Optional)
10. [x]Identification of Person(s) Making This Information Disclosure Statement

(complete the following, if appropriate)

Sections , respectively, have been continued on ADDED PAGE(S).

NOTE :"Once the minimum requirements are met, the examiner has an obligation to consider the information." Notice of April 20, 1992 (1138 O.G. 37-41, 37).

Section 1. Preliminary statements

Applicants submit herewith patents, publications or other information, of which they are aware that they believe may be material to the examination of this application, and in respect of which, there may be a duty to disclose.

The filing of this information disclosure statement shall not be construed as a representation that a search has been made (37 C.F.R. section 1.97(g)), an admission that the information cited is, or is considered to be, material to patentability, or that no other material information exists.

The filing of this information disclosure statement shall not be construed as an admission against interest in any manner. Notice of January 9, 1992, 1135 O.G. 13-25, at 25.

SECTION 2: FORMS PTO/SB/08A and 08B (formerly Form PTO-1449)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT

U.S. Patent Documents

Examiner Initials	Highlight	Ref Num.	Patent	Issue Date	Inventor	Class/ Subclass
		AA	6029168	02/22/00	Frey	707/10
		AB	5897638	04/27/99	Lasser, *et al.*	707/102
		AC	5917998	06/29/99	Cabrera, *et al.*	711/114
			5583995	12/10/96	Gardner	348/7
			5473362	12/5/95	Fitzgerald	348/7
			6775679	08/10/04	Gupta, U Day	707/102
			4993030	02/12/91	Krakauer	371/401
	*		6397246	05/28/02	Wolfe, Daniel	709/217
			6047129	04/4/00	Frye, Russell	395/712
	*		6985936	01/10/06	Agarwalla	709/221
			6044367	03/28/00	Wolff, James	707/2
	*		5649200	07/15/97	Leblang, *et al*	395/703
			6516351	02/4/03	Borr, Andrea	709/229
			6393581	05/21/02	Friedman, *et.al*	714/4
			6223206	04/21/01	Dan, *et al.*	709/105
			6161145	12/12/00	Bainbridge, *et al.*	709/246
			6339785	01/15/02	Feigenbaum, Idan	709/213
			6324581	11/27/01	Xu, *et al.*	709/229
			6438595	08/20/02	Blumenau, *et al.*	709/226
			6721794	04/13/04	Taylor	709/231
			6516350	02/14/03	Lumelsky	709/226
	*		5548724	08/20/96	Akizawa *et al.*	395/200.03
	*		6782450	08/24/04	Arnott *et al.*	711/114
	*		7072917	07/4/06	Wong *et al.*	707/205
			6233648	05/2001	Tomita, Haruo	711/4
			6556998	04/2003	Mukherjee et al.	707/10
			6985956	01/2006	Luke et al.	709/229
			6990547	01/2006	Ulrich et al.	710/304
			6990667	01/2006	Ulrich et al.	718/105

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT

U.S. Published Patent Applications

Examiner Initials	Highlight	Ref. Num	Patent	Issue Date	Inventor	Class/ Subclass
			2004/0025013	02/5/04	Parker, *et al.*	713/163
	*		2005/0021615	01/27/05	Arnott, *et al.*	709/203
	*		2004/0028043	02/12/04	Maveli, *et al.*	370/392
	*		2004/0030857	02/12/04	Krakirian, *et al.*	711/206
	*		2004/0028063	02/12/04	Roy, *et al.*	370/402
			2002/0120763	08/2002	Miloushev *et al.*	709/230
			2002/0161911	10/2002	Pinckney *et al.*	709/231
			2003/0028514	02/2003	Lord *et al.*	707/1
			2003/0033308	02/2003	Patel *et al.*	707/10
			2003/0135514	07/2003	Patel *et al.*	707/102
			2004/0006575	01/2004	Visharam *et al.*	707/104.1
			2004/0098383	05/2004	Tabellion *et al.*	707/003
			2004/0133607	07/2004	Miloushev *et al.*	707/200
			2004/0133577	07/2004	Miloushev *et al.*	707/010
			2004/0236798	11/2004	Srinivasan *et al.*	707/200

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT

Non-Patent Documents

Examiner Initials	Highlight	Ref. Num.	Author, Title of Article, Title of Journal, Volume Number, Page Numbers, Date
		AG	Callaghan *et al.*, "NFS Version 3 Protocol Specifications" (RFC 1813), 1995, The Internet Engineering Task Force (IETF), www.jetf.org, last accessed on 12/30/02.
		AH	Norton *et al.*, "CIFS Protocol Version CIFS-Spec 0.9", 2001, Storage Networking Industry Association (SNIA), www.snia.org, last accessed on 3/26/01.
		AI	Stakutis, C., "Benefits of SAN-based file system sharing", Jul. 2000, InfoStor, www.infostor.com last accessed on 12/30/02.
		AJ	Haskin *et al.*, 'The Tiger Shark File System", 1996, in proceedings of IEEE, Spring COMPCON, Santa Clara, CA, www.research.ibm.com, last accessed on 12/30/02.
		AK	Peterson, M., "Introducing Storage Area Networks", Feb 1998, InfoStor, www.infoster.com, last accessed on 12/20/02.
		AL	Patterson *et al.*, "A case for redundant arrays of inexpensive disks (RAID)", Chicago, Illinois, June 1-3, 1998, in proceedings of ACM SIGMOD conference on the Management of Data, pp.109-116, Association for Computing Machinery, Inc., www.acm.org, last accessed on 12/20/02.
		AM	Farley, M., "Building Storage Networks", January 2000, McGraw Hill, ISBN 0072120509.
		AN	"Auspex Storage Architecture Guide", Second Edition, 2001, Auspex Systems, Inc., www.auspex.com, last accessed on 12/30/02.
		AO	"Windows Clustering Technologies-An Overview", November 2000, Microsoft Corp., www.microsoft.com, last accessed on 12/30.02.
		AP	Steven Soltis, et al., 'The Global File System", in Proceedings of the Fifth NASA Goddard Space Flight Center Conference on Mass Storage Systems and Technologies, September 17-19, 1996, College Park, Maryland.
		AQ	Preslan *et al.*, "Scalability and Failure Recovery in a Linux Cluster File System", in Proceedings of the 4th Annual Linux Showcase & Conference, Atlanta, Georgia, October 10-14,2000, www.usenix.org, last accessed on 12/20/02.
		AR	Zayas, E., "AFS-3 Programmer's Reference: Architectural Overview", Transarc Corp., version 1.0 of September 2, 1991, doc.. number FS-00-D160.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT

Non-Patent Documents (Continued)

Examiner Initials	Highlight	Ref. Num.	Author, Title of Article, Title of Journal, Volume Number, Page Numbers, Date
		AS	"The AFS File System in Distibuted Computing Enviroment", Ma??96, Transarc Corp. www.transarc.ibm.com, last accessed on 12/20/02.
		AT	"VERITAS SANPoint Foundation Suite(tm) and SANPoint Foundation(tm) Suite HA: New VERITAS Volume Management and File System Technology for Cluster Environments", September 2001, VERITAS Software Corp.
		AU	"Distributed File System: Logical View of Physical Storage: White Paper", 1999, Microsoft Corp., www.microsoft.cm, last accessed on 12/20/02.
		AV	Anderson *et al.*, "Serverless Network File System", in the 15th Symposium on Operating Systems Principles, December 1995, Association for Computing Machinery, Inc.
		AW	Gibson *et al.*, "NASD Scalable Storage Systems", June 1999, USENIX99, Extreme Linux Workshop, Monterey, California.
		AX	Gibson *et al.*, "File Server Scaling with Network-Attached Secure Disks", in Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmentrics '97), 1997, Association for Computing Machinery, Inc.
		AY	Cabrera *et al.*, 'Swift: Storage Architecture for Large Objects", In Proceedings of the Eleventh IEEE Symposium on Mass Storage Systems, pages 123-128, Oct 1991.
		BA	Cabrera *et al.*, "Using Data Striping in a Local Area Network", 1992, technical report number UCSC-CRL-92-09 of the Computer & Information Sciences Department of University of California at Santa Cruz.
		BB	Long *et al.*, "Swift/RAID: A distributed RAID System", Computing Systems, vol. 7, pp. 333-359, Summer 1994.
	*	BC	Hartman, J., "The Zebra Striped Network File System", 1994, Ph.D. dissertation submitted in the Graduate Division of the University of California Berkeley.
		BD	Carns *et al.*, "PVFS: A Parallel File System For Linux Clusters", in Proceedings of the 4th Annual Linux Showcase and Conference, pages 317-327, Atlanta, Georgia, October 200, USENIX Association.
	*	BF	"NERSC Tutorials: I/O on the Cray T3E", chapter 8, "Disk Striping", National Energy Research Scientific Computing Center (NERSC), http://hpcf.nersc.gov, last accessed on 12/27/02.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Miloushev et al.
Application Number: 10/043,413
Filing Date: January 10, 2002
Title: **File Switch and Switched File System**

Group No.: 2142
Examiner: Prieto, Beatriz

LIST OF PATENTS AND PUBLICATIONS FOR APPLICANT'S INFORMATION DISCLOSURE STATEMENT

Non-Patent Documents (Continued)

Examiner Initials	Highlight	Ref. Num.	Author, Title of Article, Title of Journal, Volume Number, Page Numbers, Date
		BH	Thekkath *et al.*, 'Frangipani: A Scalable Distributed File System", in Proceedings of the 16th ACM Symposium on Operating Systems Principles, October 1997, Association for Computing Machinery, Inc.
		BI	Hwang *et al.*, Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space", IEEE Concurrency, pp. 60-69, Jan-Mar 1999.
			Cavale, M. R., "Introducing Microsoft Cluster Service (MSCS) in the Windows Server 2003," Microsoft Corporation, November 2002.
			Pearson, P.K., "Fast Hashing of Variable-Length Text Strings", Comm. Of the ACM, Vol. 33, No. 6, June 1990.
			Sorenson, K.M., "Installation and Administration: Kimberlite Cluster Version 1.1.0, Rev. D." Mission Critical Linux, http://oss.missioncriticallinux.com/kimberlite/kimberlite.pdf
			Wilkes, J., et al., "The HP AutoRAID Hierarchical Storage System," ACM Transactions on Computer Systems, Vol. 14, No. 1, February 1996.
			Savage, et al., "AFRAID- A Frequently Redundant Array of Inexpensive Disks," 1996 USENIX Technical Conf., San Diego, California, January 22-26, 1996.

Examiner: _____

Date Considered: _____

NOTE FOR EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609; draw line through citation if not in conformance AND not considered. Include copy of this form with next communication to applicant.

Section 3. Statement as to Information Not Found in Patents or Publications (Information Not Listed in Forms PTO/SB/08A and 08B (substitute for Form PTO-1449)

The following patent applications may include technically-related subject matter and/or claims that are similar to this application:

Appl. No.	File Date	Title/Description	Inventors
10/043,413	1/10/2002	File Switch and Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,704	1/2/2003	Transaction Aggregation in a Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,833	1/2/2003	Directory Aggregation for Files Distributed Over a Plurality of Servers in a Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,835	1/2/2003	Metadata Based File Switch and Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,832	1/2/2003	Rule Based Aggregation of Files and Transactions in a Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,834	1/2/2003	Aggreated Lock Management for Locking Aggregate Files in a Switched File System	Vladimir Miloushev, Peter Nickolov
10/336,784	1/2/2003	Aggregated Opportunistic Lock and Aggregated Implicit Lock Management for Locking Aggregated Files in a Switched File System	Vladimir Miloushev, Peter Nickolov
11/337,190	1/20/2006	Scalable System for Partitioning and Accessing Metadata Over Multiple Servers	Francesco Lacapra
11/041,147	1/21/2005	File-based Hybrid File Storage Scheme Supporting Multiple File Switches	Francesco Lacapra
11/072,892	3/3/2005	System and Method for Managing Small-Size Files in an Aggregated File System	Francesco Lacapra, Srinivas Duvvuri
11/285,677	11/21/2005	Directory Aggregation for Files Distributed Over a Plurality of Servers in a Switched File System	Vladimir Miloushev, Peter Nickolov
11/724,107	3/14/2007	Transaction aggregation in a switched file system	Vladimir Miloushev, Peter Nickolov

The Examiner is requested to review the entire file histories of these applications, including cited references, Office Actions, Responses, etc., and is asked to contact Applicant's Attorney if the Examiner would like the Applicant to supply copies of any or all of the information included in any of these applications. For any of these applications, if Applicant's Attorney is not contacted by the Examiner with such a request, then it will be assumed that the Examiner has reviewed or will review the file content of the application. The identification of the above-identified applications is not a waiver of secrecy for any of the applications.

Section 6. Copies of Listed Information Items Accompanying This Statement

NOTE: 37 C.F.R. section 1.98(a)(2) requires that any information disclosure statement filed under section 1.97 shall include: "A legible copy of: (1) Each U.S. and foreign patent; (ii) Each publication or that portion which caused it to be listed; and (iii) All other information or that portion which caused it to be listed, except that no copy of a U.S. patent application need be included . . ."

NOTE: The wording in section 1.98(a)(2)(iii) makes it clear that the requirement to submit a copy of each item of information listed in an information disclosure statement does not apply to the citation of a U.S. patent application. Notice of January 9, 1992, 1135 O.G. 13-25, at 14.

Legible copies of all items listed in Forms PTO/SB/08A and 08B (substitute for Form PTO-1449) accompany this information statement.

(complete the following, if applicable)

[x]Exception(s) to above:

U.S. patent citations are not included pursuant to the United State Patent and Trademarks Office's September 21, 2004 waiver of the copy requirement in 37 CFR 1.98 for cited pending U.S. patent citations when the patent citations are available in the USPTO's IFW system.

[ ]Items in prior application, from which an earlier filing date is claimed for this application, as identified in Section 4.

[ ]Cumulative patents or publications identified in Section 5.

Section 9. Concise Explanation of English Language Listed Information Items

NOTE: "Applicants may, if they wish, provide a concise explanation of why English-language information is being submitted and how it is understood to be relevant. Concise explanations are helpful to the Office, particularly where documents are lengthy and complex and applicant is aware of a section that is highly relevant to patentability or where a large number of documents are submitted and applicant is aware that one or more are highly relevant to patentability." Notice of April 20, 1992 (1138 O.G. 37-47, 38).

Applicants appreciate that the disclosure of such a large number of references may be burdensome on the Examiner. Most of the references have been cited either by the USPTO or by the Applicants in one or more of the related applications listed in Section 3 above, and therefore the Examiner may find such references to be relevant to examination of the subject patent application. References that were not cited in one or more of those related applications are among those highlighted with an asterisk (*) next to the reference in the column labeled "Highlight." Applicants are not submitting these references in an attempt to prolong prosecution or hide information that may be material to patentability. Applicants' attorney has reviewed the references and believes that none is more relevant to the subject patent application than the references already of record. Nevertheless, Applicants request that the Examiner perform an independent review of the references. In an attempt to facilitate the Examiner's review of the references, Applicants' attorney would direct the Examiner's attention to the highlighted references, which, out of the cited references, appear to be more closely related to the subject patent application.

Submission of any particular reference is not an admission that the reference is material to patentability or qualifies as prior art to one or more of the claims.

Section 10. Identification of Person(s) Making This Information Disclosure Statement

The person making this certification is

(check each applicable item)

(a) [ ]the inventor(s) who signs below

SIGNATURE OF INVENTOR

(type name of inventor who is signing)

(b) [ ]an individual associated with the filing and prosecution of this application (37 C.F.R. section 1.56(c))

SIGNATURE OF INVENTOR

(type name of inventor who is signing)

(c) [x] the practitioner who signs below on the basis of the information:

(check each applicable item)

[ ] supplied by the inventor(s).

[ ] supplied by an individual associated with the filing and prosecution of this application. (37 C.F.R. section 1.56(c)).

[x] in the practitioner's file.

SIGNATURE OF PRACTITIONER

Reg. No.: 39,250

Jeffrey T. Klayman
(type or print name of practitioner)

Tel. No.: (617) 443-9292

Bromberg & Sunstein LLP
125 Summer Street, 11th Floor
P.O. Address

Customer No.: 02101

Boston, MA 02110

03193/00102 764435.1

RFC 1813

RFC Archive

Network Working Group
Request for Comments: 1813
Category: Informational
Sun Microsystems, Inc.
June 1995

B. Callaghan
B. Pawlowski
P. Staubach

NFS Version 3 Protocol Specification

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

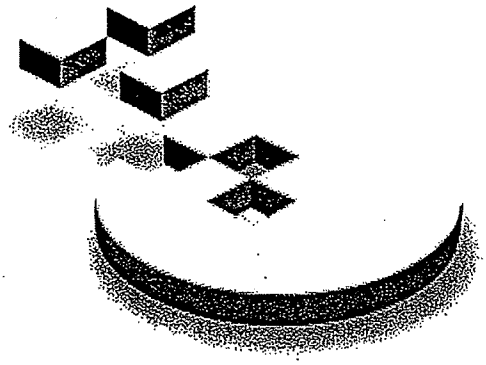
IESG Note

Internet Engineering Steering Group comment: please note that the IETF is not involved in creating or maintaining this specification. This is the significance of the specification not being on the standards track.

Abstract

This paper describes the NFS version 3 protocol. This paper is provided so that people can write compatible implementations.

Table of Contents



SNIA

Storage Networking Industry Association

Common Internet File System (CIFS)

Version: CIFS-Spec 0.9

SNIA CIFS Documentation Work Group
Contact: snia-cifsspec@snia.org

Draft SNIA CIFS Work Group Work-in-Progress

Revision Date: 3/26/2001 6:10 PM

Current Status: Preliminary Work Group Review
NOT FOR DEVELOPMENT USAGE

INFOSTOR

Benefits of SAN-based file system sharing

Software that allows users to share file systems in storage area network environments may reduce total cost of ownership.

■ BY CHRIS STAKUTIS

Storage area networks (SANs) have promised many improvements in managing and utilizing data. Certainly, co-locating and centralizing storage at the physical level helps with overall cost and management. And Logical Unit Number (LUN) masking can be used to partition storage into logical pieces, assigned to different servers. However, LUN masking is only a first step in realizing the full potential of SANs. Additional benefits are possible by sharing file systems and data among simultaneously connected servers.

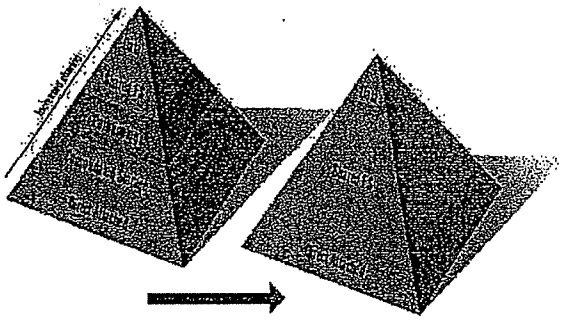


FIGURE 1. Shared file systems combine the advantages of other sharing techniques.

There are a number of products in the SAN market that offer the ability to share file systems. The main challenge the industry currently faces is not the technical implementation of shared file systems but, rather, educating the end-user community about these solutions and how they add value to SAN implementations.

Technology advances have allowed a continuous progression of sharing in a SAN environment (see Figure 1). Until recently, the diagram on the left in Fig. 1 represents how most enterprise managers looked at SANs. Now, with the realization of the true value of a shared SAN file system, the diagram on the right is more accurate. The fundamental difference is in realizing how and when to deploy a shared SAN file system, and for what benefit.

Shared file system SANs have been in existence for several years. Historically, users have concentrated on the top part of the shared-access pyramid; that is, the typical user has a pressing need for every computer to share the exact same data files at nearly the same time. Examples include movie and video production, digital graphics and pre-press, and other vertical markets such as medical and geo-imaging. However, there is a wider

There are a number of SAN file systems available today, and there are two prevailing architectures.

Reprinted from the July 2000 edition of InfoStor
Copyright 2000 by PennWell Corporation

IBM

Search

Home Products & services Support & downloads My account

Select a country

Journals Home
Systems Journal
Journal of Research and Development
- Current Issue
- Recent Issues
- Papers in Progress
- Search/Index
- Orders
- Description
- Patents
- Recent publications
- Author's Guide

Staff
Contact Us

IBM Journal of Research and Development
Volume 42, Number 2, 1998
MultiMedia Systems

Table of contents: HTML ASCII This article: HTML ASCII DOI: 10.1147/rd.422.0185 Copyright info

Tiger Shark--A scalable file system for multimedia

by R. L. Haskin

Tiger Shark is a scalable, parallel file system designed to support interactive multimedia applications, particularly large-scale ones such as interactive television (ITV). Tiger Shark runs under the IBM AIX® operating system, on machines ranging from RS/6000™ desktop workstations to the SP2® parallel supercomputer. In addition to supporting continuous-time data, Tiger Shark provides scalability, high availability, and on-line system management, all of which are crucial in large-scale video servers. These latter features also enable Tiger Shark to support nonmultimedia uses, such as scientific computing, data mining, digital library, and scalable network file servers. Tiger Shark has been employed in a number of customer ITV trials. On the basis of experience obtained from these trials, Tiger Shark has recently been released in several IBM video-server products. This paper describes the architecture and implementation of Tiger Shark, discusses the experience gained from trials, and compares Tiger Shark to other scalable video servers.

Introduction

To date, most multimedia application programs run on stand-alone personal computers, with digitized video and audio coming from local hard disks and CD-ROMs. Increasingly, there has been a demand for file servers for multimedia data. The reasons for this include those that motivate the use of file servers for conventional data: sharing, security, and centralized administration.

It is difficult for a conventional file server to handle multimedia data. When a conventional file server becomes overloaded, all users experience lower throughput and greater response time. For multimedia, the file server must deliver digitized video and/or audio data at a rate that allows it to be presented to the user in a smooth, continuous stream (this is called continuous-time, or *isochronous,* presentation). Any nontrivial delay by the server results in *stream starvation,* which appears to the user as an annoying interruption in the presentation. Stream starvation can be avoided by buffering data and/or underloading the file server, but either of these alternatives can increase cost prohibitively. A video server differs from a conventional file

INFOSTOR

PRINTTHIS
EMC²
where information lives

InfoStor QuickVote
FOR END USERS, ONLY: Do you plan to:

Vote Now!

Introducing Storage Area Networks

Introducing Storage Area Networks

Here's a primer to get you started in understanding the benefits of SAN, SAS, and NAS architectures.

By Michael Peterson

As IT organizations re-engineer distributed networks to achieve continuous operations and to host mission-critical applications, they are increasingly considering an architecture that is common in data centers. Mainframe-based data centers use a network storage interface called ESCON to connect mainframes to multiple storage systems and distributed networks, an architecture that is referred to as a storage-area network (SAN).

In a typical data center, SANs account for approximately 25% of all network traffic. What's new is that SAN architectures are now being adopted in distributed networks using low-cost interconnect technologies such as SCSI, SSA, and Fibre Channel.

A SAN is a high-speed network, similar to a LAN, that establishes a direct connection between storage elements and servers or clients. The SAN is an extended storage bus that can be interconnected using technologies used in LANs and WANs, such as routers, hubs, switches, and gateways. A SAN can be local or remote, shared or dedicated, and it uniquely includes externalized and central storage. SAN interfaces are usually ESCON, SCSI, SSA, Fibre Channel, or HIPPI, rather than Ethernet.

It doesn't matter whether a SAN is called a storage-area network or system-area network because the architecture is the same. Either way, SANs create a method of attaching storage that is revolutionizing networks, resulting in significant improvements in availability and performance.

SANs are currently used to connect shared-storage arrays, to cluster servers for failover, to interconnect mainframe disk or tape resources to distributed network servers and clients, and to create parallel or alternate data paths for high-performance computing environments.

In essence, a SAN is nothing more than another network, like a subnet, except that it's implemented with storage interfaces. SANs enable storage to be externalized from the server, allowing storage devices to be shared among multiple host servers without affecting system performance or the primary network.

SANs are not new. The benefits are well proven because the architecture evolved from mainframe DASD. In fact, Digital Equipment's VAX/VMS network environment is based on a SAN architecture and clustered servers. And vendors such as EMC already have a large installed base of SAN-attached arrays.

PORTAL

> home > about > feedback > login

Citation

International Conference on Management of Data and Symposium on Principles of Database Systems >archive

Proceedings of the 1988 ACM SIGMOD international conference on Management of data >toc

1988 , Chicago, Illinois, United States

A case for redundant arrays of inexpensive disks (RAID)

Authors

David A. Patterson Univ. of California, Berkeley

Garth Gibson Univ. of California, Berkeley

Randy H. Katz Univ. of California, Berkeley

Sponsor

SIGMOD : ACM Special Interest Group on Management of Data

Publisher

ACM Press New York, NY, USA

Pages: 109 - 116 Series-Proceeding-Article

Year of Publication: 1988

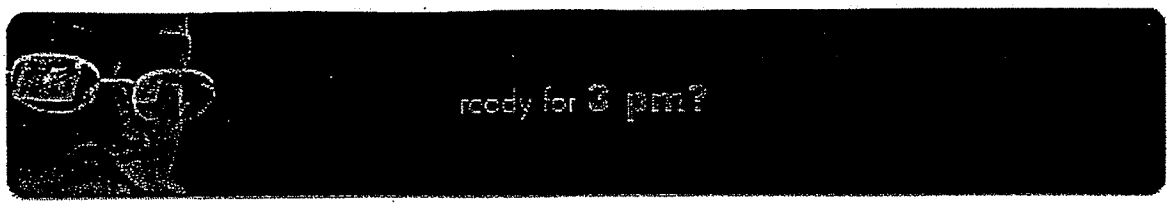
ISBN:0-89791-268-3

doi> http://doi.acm.org/10.1145/50202.50214 (Use this link to Bookmark this page)

> full text > abstract > references > citings > index terms > peer to peer

> Discuss > Similar > Review this Article Save to Binder

> BibTex Format



Enterprise Storage Forum

Back to Article

Book Review - Building Storage Networks, 2nd Edition

By Enterprise Storage Forum Staff

Building Storage Networks (2nd Ed) by Marc Farley is nothing if not complete. At almost 600 pag has the capacity to deliver a wide reaching range of material on storage network which it duly doe:

Farley, who also wrote the first edition of the book, makes it clear in the introduction that much ha changed both in the storage industry, and since the book's first edition. He also explains that while has included information about emerging technologies such as Infiniband, there is plenty that can change as these nascent technologies develop. Farley is to be commended in this respect, as it wou have been easy for him to skimp on coverage of such topics. Instead, he has obviously expended a deal of time and effort in researching new technologies and including the information in the book.

The seventeen (yes, that's one seven) chapters of the book take the reader on a progressive journey through almost every aspect of networked storage. There are five sections within the book which f a natural progression - Introduction to Network Storage, Fundamental Storing Applications, The Storage Channel Becomes a Network, Wiring Technologies and Filing, Internet Storage, and Management.

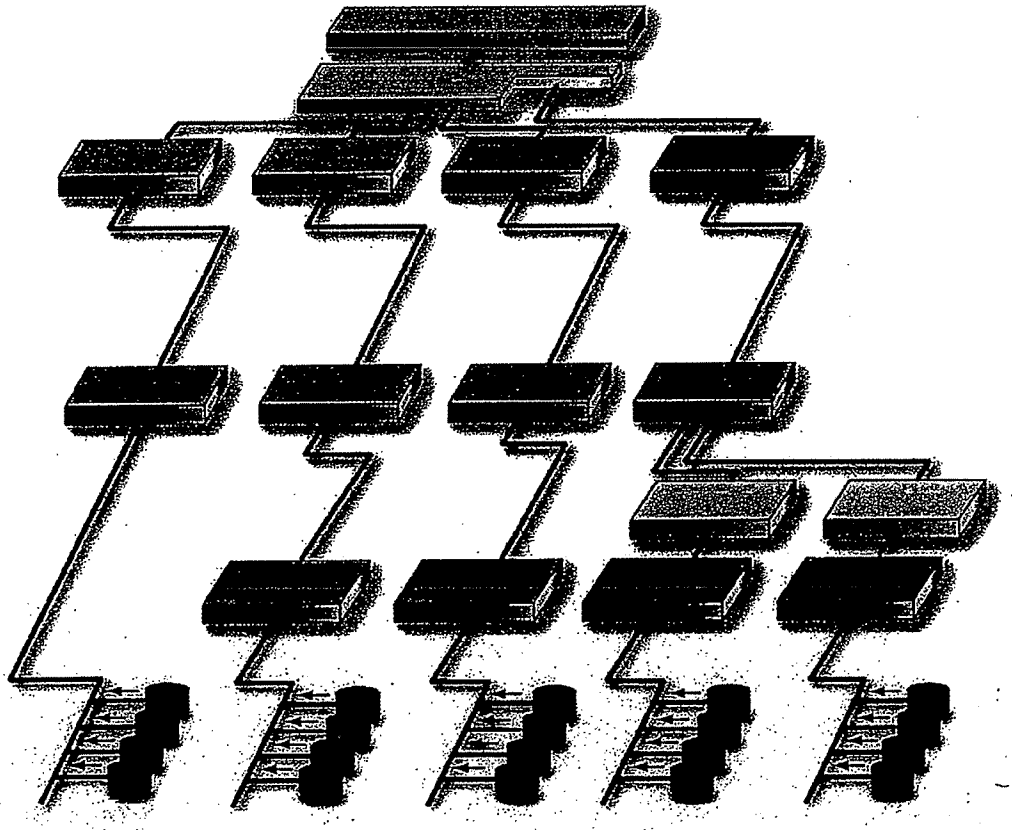
Perhaps the most impressive aspect of the book is the sheer detail used to explain the underlying principles of network storage before progressing to the more common coverage of SAN's, NAS an Fibre Channel. In fact, by the time the detailed discussion on SAN's and NAS is reached, the reade have already covered RAID, caching, backup, I/O channels, mirroring and replication, and networ backup. Detailed coverage on SANs doesn't start until page 257 by which time the reader is in a st position to better understand why SAN's are used and of the underlying technologies.

In the middle of the book, an 8 page 'blueprint' section runs down, in graphical form, some of the 'Typical Implementations of Filing, Storing, and Wiring in Host Systems and Storage Subsystems. Figures abound throughout the book and do an excellent job of reinforcing the principles describe the text. A large number of them make use of flow-chart type shapes whereas others, when approp use representations of servers, hubs and other network storage paraphernalia.

One slightly odd inclusion is that of exercises at the end of each section, which incite the reader to perform tasks like "Diagram the complete process and I/O path used in retrieving data from virtual memory on disk." While such an exercise might be a good reinforcement of the information prese in the chapter, these tasks seem more suited to a certification study guide than a reference book.

A Storage Architecture Guide - Second Edition

A business and technical discussion of the integration of NAS and SAN



AUSPEX

Auspex Technical Report

Windows Clustering Technologies—An Overview

Published: November 2001

Abstract

This article is written for IT managers and examines the cluster technologies available on the Microsoft® Windows® server operating system. Also discussed is how cluster technologies can be architected to create comprehensive, mission-critical solutions that meet the requirements of the enterprise.

The Global File System[1]

Steven R. Soltis, Thomas M. Ruwart, Matthew T. O'Keefe
Department of Electrical Engineering
and
Laboratory for Computational Science and Engineering
University of Minnesota
4-174 EE/CS Building
Minneapolis, MN 55455
soltis@ee.umn.edu
Tel: +1-612-625-6306
Fax: +1-612-625-4583

Abstract

The Global File System (GFS) is a prototype design for a distributed file system in which cluster nodes physically share storage devices connected via a network-like Fibre Channel. Networks and network-attached storage devices have advanced to a level of performance and extensibility so that the previous disadvantages of *shared disk* architectures are no longer valid. This shared storage architecture attempts to exploit the sophistication of storage device technologies whereas a server architecture diminishes a device's role to that of a simple component. GFS distributes the file system responsibilities across processing nodes, storage across the devices, and file system resources across the entire storage pool. GFS caches data on the storage devices instead of the main memories of the machines. Consistency is established by using a locking mechanism maintained by the storage devices to facilitate atomic read-modify-write operations. The locking mechanism is being prototyped on Seagate disk drives and Ciprico disk arrays. GFS is implemented in the Silicon Graphics IRIX operating system and is accessed using standard Unix commands and utilities.

Introduction

Distributed systems can be evaluated by three factors: performance, availability, and extensibility. Performance can be characterized by such measurements as response time and throughput. Distributed systems can achieve availability by allowing their working components to act as replacements for failed components. Extensibility is a combination of portability and scalability. Obvious influences on scalability are such things as addressing limitations and network ports, but subtle bottlenecks in hardware and software may also arise.

These three factors are influenced by the architecture of the distributed and parallel systems. The architectures can be categorized as *message-based* (*shared nothing*) and

[1] This work was supported by the Office of Naval Research under grant no. N00019-95-1-0611, by the National Science Foundation under grant ASC-9523480, and by grant no. 5555-23 from the University Space Research Association which is administered by NASA's Center for Excellence in Space Data and Information Sciences (CESDIS) at the NASA Goddard Space Flight Center.

4th Annual Linux Showcase & Conference, Atlanta

Pp. 169–180 of the *Proceedings*

Scalability and Failure Recovery in a Linux Cluster File System

Kenneth W. Preslan, Andrew Barry, Jonathan Brassow, Michael Declerck, A.J. Lewis, Adam Manthei, Ben Marzinski, Erling Nygaard, Seth Van Oort, David Teigland, Mike Tilstra, Steven Whitehouse, and Matthew O'Keefe

Sistina Software, Inc.
1313 5th St. S.E.
Minneapolis, Minnesota 55414
+1-612-379-3951, okeefe@sistina.com

Abstract:

In this paper we describe how we implemented journaling and recovery in the Global File System (GFS), a shared-disk, cluster file system for Linux. We also present our latest performance results for a 16-way Linux cluster.

Introduction

Traditional local file systems support a persistent name space by creating a mapping between blocks found on disk drives and a set of files, file names, and directories. These file systems view devices as local: devices are not shared so there is no need in the file system to enforce device sharing semantics. Instead, the focus is on aggressively caching and aggregating file system operations to improve performance by reducing the number of actual disk accesses required for each file system operation [1], [2].

AFS-3 Programmer's Reference: Architectural Overview

Edward R. Zayas

Transarc Corporation

Version 1.0 of 2 September 1991 22:53
©Copyright 1991 Transarc Corporation
All Rights Reserved
FS-00-D160

The AFS File System In Distributed Computing Environments

Introduction

The AFS 3 distributed file system targets the issues critical to distributed computing environments. AFS performs exceptionally well, both within small, local work groups of machines and across wide-area configurations in support of large, collaborative efforts. AFS provides an architecture geared towards system management, along with the tools to perform important management tasks. For a user, AFS is a familiar yet extensive UNIX environment for accessing files easily and quickly.

AFS 3 is currently in use at hundreds of sites worldwide, with the number of AFS sites continuing to grow at a robust rate. Looking to the future, the Open Software Foundation's (OSF) Distributed Computing Environment (DCE) includes a Distributed File Service (DFS) based on AFS 3. Along with the DCE, the DFS will be provided to a multitude of new sites by way of major hardware vendors and other third party developers (including Transarc).

This document discusses the attributes of AFS 3 and how it is used today in client/server environments. The document compares AFS 3 to the Network File System (NFS), explaining the advantages of AFS over an NFS environment. An additional topic is the use of AFS 3 as a stepping stone to the DCE, providing both a learning environment and a way to migrate existing resources.

AFS 3 Attributes

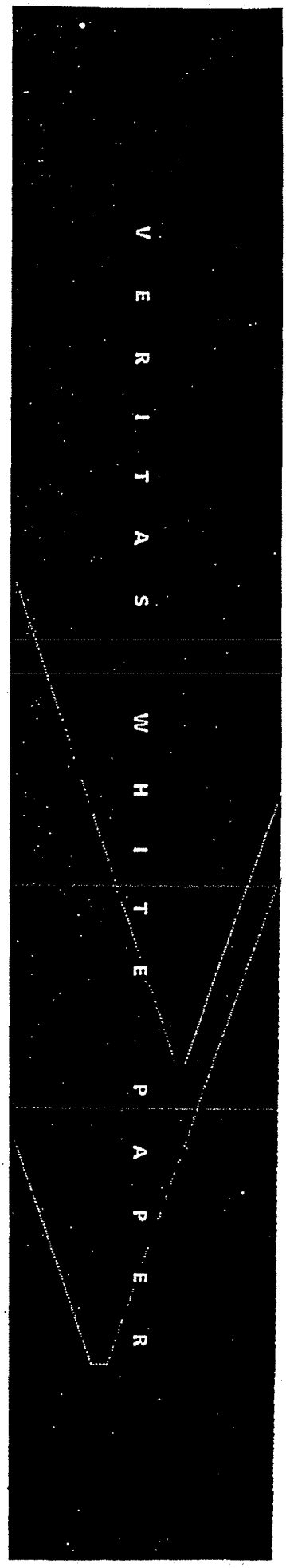
A single, shared name space for all users, from all machines. AFS brings together all of the files stored within the file system into a single name space. Every AFS user shares this same name space, making all AFS files easily available from any AFS machine.

Location-independent file sharing. With AFS, the name of a file is independent of both the file's and the user's physical location, contributing to ease of file sharing and resource management.

Client caching and efficient wide-area protocols for excellent performance. Both small and large-scale distributed environments benefit from AFS mechanisms to reduce server and network load. AFS caches data on client machines to reduce subsequent data requests directed at file servers, substantially reducing network and server loads. Servers keep track of client caches through callbacks, so a client does not need to constantly query the server to see if the file has changed.

The AFS remote procedure call reads and writes data to a remote procedure call (RPC) stream, further improving the efficiency of data transfer across a local- or

VERITAS



VERITAS SANPoint Foundation Suite™ and SANPoint Foundation Suite™ HA

New VERITAS Volume Management and File System Technology for Cluster Environments

September 2001

Distributed File System: A Logical View of Physical Storage

White Paper

Abstract

The Microsoft® Distributed File System (Dfs) is a network server component that makes it easier for you to find and manage data on your network. Dfs is a means for uniting files on different computers into a single name space. Dfs makes it easy to build a single, hierarchical view of multiple file servers and file server shares on your network.

Microsoft Distributed File System version 4.1 for Microsoft Windows NT® Server 4.0 is currently available for download from the Microsoft Web site at http://www.microsoft.com/ntserver . This release includes the Microsoft Windows® 95 operating system Dfs client and enhanced security signatures. In addition, Microsoft Windows® 2000 will include directory service–enabled enhancements to Dfs. This paper covers Dfs technology as a whole, including the version for Windows 2000.

Introduction

The Distributed File System (Dfs) for the Microsoft® Windows NT® Server and Microsoft Windows® 2000 Server operating systems is a network server component that makes it easier for you to find and manage data on your network. Dfs is a means for uniting files on different computers into a single name space. Dfs makes it easy to build a single, hierarchical view of multiple file servers and file server shares on your network. Instead of seeing a physical network of dozens of file servers, each with a separate directory structure, users will now see a few logical directories that include all of the important file servers and file server shares. Each share appears in the most logical place in the directory, no matter what server it is actually on.

Dfs does for servers and shares what file systems do for hard disks. File systems provide uniform named access to collections of sectors on disks; Dfs provides a uniform naming convention and mapping for collections of servers, shares, and files. Thus, Dfs makes it possible to organize file servers and their shares into a logical hierarchy, making it considerably easier for a large corporation to manage and use its information resources. In addition, Dfs is not limited to a single file protocol and can support the mapping of servers, shares, and files, regardless of the file client being used, provided that the client supports the native server and share.

What is a Distributed File System?

Dfs provides name transparency to disparate server volumes and shares. Through Dfs, an administrator can build a single hierarchical file system whose contents are distributed throughout your organization's WAN. In short, Dfs can be thought of as a share of other shares.

Historically, with the universal naming convention (UNC), a user or application was required to specify the physical server and share in order to access file information (that is, the user or application had to specify *\\Server\Share\Path\Filename*). Even though UNCs can be used directly, a UNC is typically mapped to a drive letter where *x:* might be mapped to *\\Server\Share*. From that point, a user had to navigate beyond the redirected drive mapping to the data he or she wishes to access (for example, copy *x:\Path\More_ path\.....\Filename*.

As networks continue to grow in size and as organizations begin to use existing storage—both internally and externally—for purposes such as intranets, mapping a single drive letter to individual shares scales poorly. Further, although users can use UNC names directly, these users can be overwhelmed by the number of places where data can be stored.

Dfs solves these problems by permitting the linking of servers and shares into a simpler, more meaningful name space. This new Dfs volume permits shares to be hierarchically connected to other Windows shares. Since Dfs maps the physical storage into a logical representation, the net benefit is that the physical location of data becomes transparent to users and applications.

Benefits of Dfs

Serverless Network File Systems

Thomas E. Anderson, Michael D. Dahlin, Jeanna M. Neefe,
David A. Patterson, Drew S. Roselli, and Randolph Y. Wang

Computer Science Division
University of California at Berkeley

Abstract

In this paper, we propose a new paradigm for network file system design, *serverless network file systems*. While traditional network file systems rely on a central server machine, a serverless system utilizes workstations cooperating as peers to provide all file system services. Any machine in the system can store, cache, or control any block of data. Our approach uses this location independence, in combination with fast local area networks, to provide better performance and scalability than traditional file systems. Further, because any machine in the system can assume the responsibilities of a failed component, our serverless design also provides high availability via redundant data storage. To demonstrate our approach, we have implemented a prototype serverless network file system called xFS. Preliminary performance measurements suggest that our architecture achieves its goal of scalability. For instance, in a 32-node xFS system with 32 active clients, each client receives nearly as much read or write throughput as it would see if it were the only active client.

1. Introduction

A serverless network file system distributes storage, cache, and control over cooperating workstations. This approach contrasts with traditional file systems such as Netware [Majo94], NFS [Sand85], Andrew [Howa88], and Sprite [Nels88] where a central server machine provides all file system services. Such a central server is both a performance and reliability bottleneck. A serverless system, on the other hand, distributes control processing and data storage to achieve scalable high performance, migrates the responsibilities of failed components to the remaining machines to provide high availability, and scales gracefully to simplify system management.

Three factors motivate our work on serverless network file systems: the opportunity provided by fast switched LANs, the expanding demands of users, and the fundamental limitations of central server systems.

The recent introduction of switched local area networks such as ATM or Myrinet [Bode95] enables serverlessness by providing aggregate bandwidth that scales with the number of machines on the network. In contrast, shared media networks such as Ethernet or FDDI allow only one client or server to transmit at a time. In addition, the move towards low latency network interfaces [vE92, Basu95] enables closer cooperation between machines than has been possible in the past. The result is that a LAN can be used as an I/O backplane, harnessing physically distributed processors, memory, and disks into a single system.

Next generation networks not only enable serverlessness, they require it by allowing applications to place increasing demands on the file system. The I/O demands of traditional applications have been increasing over time [Bake91]; new applications enabled by fast networks — such as multimedia, process migration, and parallel processing — will further pressure file systems to provide increased performance. For instance, continuous media workloads will increase file system demands; even a few workstations simultaneously running video applications would swamp a traditional central server [Rash94]. Coordinated Networks of Workstations (NOWs) allow users to migrate jobs among many machines and also permit networked workstations to run parallel jobs [Doug91, Litz92, Ande95]. By increasing the peak processing power available to users, NOWs increase peak demands on the file system [Cyph93].

Unfortunately, current centralized file system designs fundamentally limit performance and availability since all read misses and all disk writes go through the central server. To address such performance limitations, users resort to costly schemes to try to scale these fundamentally unscalable file systems. Some installations rely on specialized server machines configured with multiple processors, I/O channels, and I/O processors. Alas, such machines cost significantly more than desktop workstations for a given amount of computing or I/O capacity. Many installations also attempt to achieve scalability by distributing a file system among multiple servers by partitioning the directory tree. This approach only moderately improves scalability because its coarse distribution often results in hot spots when the partitioning allocates heavily used files and directory trees to a single server [Wolf89]. It is also expensive, since it requires the (human) system manager to effectively become *part* of the file system — moving users, volumes, and disks among servers to balance load. Finally, AFS [Howa88] attempts to improve scalability by caching data on client

This work is supported in part by the Advanced Research Projects Agency (N00600-93-C-2481, F30602-95-C-0014), the National Science Foundation (CDA 0401156), California MICRO, the AT&T Foundation, Digital Equipment Corporation, Exabyte, Hewlett Packard, IBM, Siemens Corporation, Sun Microsystems, and Xerox Corporation. Anderson was also supported by a National Science Foundation Presidential Faculty Fellowship, Neefe by a National Science Foundation Graduate Research Fellowship, and Roselli by a Department of Education GAANN fellowship. The authors can be contacted at {tea, dahlin, neefe, patterson, drew, rywang}@CS.Berkeley.EDU.

Copyright © 1995 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that new copies bear this notice and the full citation on the first page. Copyrights for components of this WORK owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specificpermission and/or a fee. Request Permissions from Publications Dept, ACM Inc.

This work first appeared in the 15th Symposium on Operating Systems Principles, December, 1995.

NASD Scalable Storage Systems

Garth A. Gibson*, David F. Nagle†, William Courtright II*, Nat Lanza*,
Paul Mazaitis*, Marc Unangst*, Jim Zelenka*

School of Computer Science*
Department of Electrical and Computer Engineering†
Carnegie Mellon University, Pittsburgh, PA 15213
www.pdl.cs.cmu.edu

ABSTRACT

The goal of CMU's Network-Attached Secure Disks (NASD) project is to define the next era of storage system interfaces and architectures. To encourage industry standardization of a compliant storage device/subsystem interface, we are working closely with the National Storage Industry Consortium's working group on network-attached storage. Our experimental demonstration of the NASD interface's value is device and filesystem prototype software that delivers the scalability inherent in a NASD storage architecture. To engage the academic community and to provide a reference implementation for industry development, CMU is releasing its Linux and Digital UNIX ports of this software. In this paper, we overview the NASD scalable storage architecture and the code-base we are releasing for Linux.

1. INTRODUCTION

Demands for storage throughput continue to grow due to ever larger clusters sharing storage, rapidly increasing client performance, richer data types such as video, and data-intensive applications such as data mining. For storage subsystems to deliver scalable throughput, that is, linearly increasing application bandwidth and accesses per second with increasing numbers of storage devices and client processors, the data must be striped over many disks and network links [Patterson88], and name lookup and access rights checking must be decentralized [Hartman93, Anderson96]. With current technology, most office, engineering, and data processing shops have sufficient numbers of disks and scalable switched networking, but they access storage through storage controller and distributed fileserver bottlenecks. These bottlenecks arise because a single "server" computer copies data between the storage (peripheral) network and the client (local area) network while adding functions such as concurrency control and metadata consistency.

Our prior work proposed a new scalable-bandwidth storage architecture, Network-Attached Secure Disks (NASD) [Gibson97a, Gibson97b, Gobioff97, Gibson98, Amiri99, Nagle99]. Fundamentally, NASD minimizes server-based data movement by separating management and filesystem semantics from store-and-forward copying and elevating commodity storage's interface to a richer object-based model (SCSI4 perhaps).

As with earlier generations of SCSI, the NASD interface is simple, efficient and flexible enough to support a wide range of filesystem semantics across multiple generations of technology. Of course, advancing storage interfaces and architecture requires industry collaboration and standardization. Fortunately, the storage industry is aggressively seeking to evolve their marketplace [Quantum99, Seagate99]. To promote network-attached storage, CMU is working closely with the National Storage Industry Consortium's (NSIC) working group on network-attached storage devices (www.nsic.org/nasd). Over the past three years, NSIC has hosted about a dozen public workshops where academics and practitioners exchange perspectives on next generation storage. Currently, the core NSIC working group is engaged in developing an ANSI standards proposal for a new storage interface.

Until recently, CMU publications have been sufficient for collaboration in the NSIC effort. Now, to more widely disseminate our work, CMU is providing, for public use, a reference implementation of NASD for the Linux 2.2 and Digital UNIX 3.2 environments. Our reference implementation includes NASD device code (running on a workstation or PC masquerading as a subsystem or disk drive), an NFS-like distributed file system designed to use NASD subsystems or devices, and NASD-inspired striping middleware to provide scalable bandwidth to large striped files. The rest of this extended abstract describes this prototype software and summarizes prior research predictions for its performance.

2. BACKGROUND AND RELATED WORK

Figure 1 illustrates the principal network-attached storage architectures. The simplest implementation runs on a standalone server with attached disks (SAD), as shown in Figure 1a. Data makes two network trips on its way to the client, making the server a potential bottleneck; particularly since a server usually manages a large numbers of disks to amortize cost. Companies such as Network Appliance have improved the performance of SAD implementations, specifically the number of clients supported, by using special purpose server hardware and highly optimized software (SID) [Hitz94].

Appears in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (Sigmetrics '97)*, Seattle, Washington, June 15-18, 1997.

File Server Scaling with Network-Attached Secure Disks

Garth A. Gibson†, David F. Nagle*, Khalil Amiri*, Fay W. Chang†, Eugene M. Feinberg*, Howard Gobioff†, Chen Lee†, Berend Ozceri*, Erik Riedel*, David Rochberg†, Jim Zelenka†

*Department of Electrical and Computer Engineering
†School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890
garth+nasd@cs.cmu.edu
http://www.cs.cmu.edu/Web/Groups/NASD/

Abstract

By providing direct data transfer between storage and client, network-attached storage devices have the potential to improve scalability for existing distributed file systems (by removing the server as a bottleneck) and bandwidth for new parallel and distributed file systems (through network striping and more efficient data paths). Together, these advantages influence a large enough fraction of the storage market to make commodity network-attached storage feasible. Realizing the technology's full potential requires careful consideration across a wide range of file system, networking and security issues. This paper contrasts two network-attached storage architectures—(1) Networked SCSI disks (NetSCSI) are network-attached storage devices with minimal changes from the familiar SCSI interface, while (2) Network-Attached Secure Disks (NASD) are drives that support independent client access to drive object services. To estimate the potential performance benefits of these architectures, we develop an analytic model and perform trace-driven replay experiments based on AFS and NFS traces. Our results suggest that NetSCSI can reduce file server load during a burst of NFS or AFS activity by about 30%. With the NASD architecture, server load (during burst activity) can be reduced by a factor of up to five for AFS and up to ten for NFS.

1 Introduction

Users are increasingly using distributed file systems to access data across local area networks; personal computers with hundred-plus MIPS processors are becoming increasingly affordable; and the sustained bandwidth of magnetic disk storage is expected to exceed 30 MB/s by the end of the decade. These trends place a pressing need on distributed file system architectures to provide clients with efficient, scalable, high-bandwidth access to stored data. This paper discusses a powerful approach to fulfilling this need. Network-attached storage provides high bandwidth by directly attaching storage to the network, avoiding file server store-and-forward operations and allowing data transfers to be striped over storage and switched-network links.

The principal contribution of this paper is to demonstrate the potential of network-attached storage devices for penetrating the markets defined by existing distributed file system clients, specifically the Network File System (NFS) and Andrew File System (AFS) distributed file system protocols. Our results suggest that network-attached storage devices can improve overall distributed file system cost-effectiveness by offloading disk access, storage management and network transfer and greatly reducing the amount of server work per byte accessed.

We begin by charting the range of network-attached storage devices that enable scalable, high-bandwidth storage systems. Specifically, we present a taxonomy of network-attached storage — server-attached disks (SAD), networked SCSI (NetSCSI) and network-attached secure disks (NASD) — and discuss the distributed file system functions offloaded to storage and the security models supportable by each.

With this taxonomy in place, we examine traces of requests on NFS and AFS file servers, measure the operation costs of commonly used SAD implementations of these file servers and develop a simple model of the change in manager costs for NFS and AFS in NetSCSI and NASD environments. Evaluating the impact on file server load analytically and in trace-driven replay experiments, we find that NASD promises much more efficient file server offloading in comparison to the simpler NetSCSI. With this potential benefit for existing distributed file server markets, we conclude that it is worthwhile to engage in detailed NASD implementation studies to demonstrate the efficiency, throughput and response time of distributed file systems using network-attached storage devices.

In Section 2, we discuss related work. Section 3 presents our taxonomy of network-attached storage architectures. In Section 4, we describe the NFS and AFS traces used in our analysis and replay experiments and report our measurements of the cost of each server operation in CPU cycles. Section 5 develops an analytic model to estimate the potential scaling offered by server-offloading in NetSCSI and NASD based on the collected traces and the measured costs of server operations. The trace-driven replay experiment and the results are the subject of Section 6. Finally, Section 7 presents our conclusions and discusses future directions.

This research was sponsored by DARPA/ITO through ARPA Order D306 under contract N00174-96-0002 and in part by an ONR graduate fellowship. The project team is indebted to generous contributions from the member companies of the Parallel Data Consortium: Hewlett-Packard, Symbios Logic Inc., Data General, Compaq, IBM Corporation, EMC Corporation, Seagate Technology, and Storage Technology Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any supporting organization or the U.S. Government.

© 1997 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that new copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted.

To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request Permissions from Publications Dept, ACM Inc. Fax +1 (212) 869-0481, or <permissions@acm.org>.

Swift: A Storage Architecture for Large Objects

Luis-Felipe Cabrera
Computer Science Department
IBM Almaden Research Center

Internet: cabrera@ibm.com

Darrell D. E. Long
Computer & Information Sciences
University of California at Santa Cruz

Internet: darrell@sequoia.ucsc.edu

Abstract

Managing large objects with high data-rate requirements is difficult for current computing systems. We describe an Input/Output architecture, called Swift, that addresses the problem of storing and retrieving very large data objects from slow secondary storage at very high data-rates. Applications that require this capability are poorly supported in current systems, even though they are made possible by high-speed networks. These range from storage and visualization of scientific computations to recodring and play-back of color video in real-time. Swift addresses the problem of providing the data rates required by digital video by exploiting the available interconnection capacity and by using several slower storage devices in parallel.

We have done two studies to validate the Swift architecture: a simulation study and an Ethernet-based proof-of-concept implementation. Both studies indicate that the aggregation principle proposed in Swift can yield very high data-rates. We present a brief summary of these studies.

1 Introduction

The disparity between processing speed, network transfer rates, and the performance of disk storage systems will increase in the future. The processing speed of computing systems continues to increase at an exponential rate. Advances in communications technology are providing increased transfer rates even more rapidly than the increases in processing speed.

In contrast to these advances, disk storage technology remains much the same. Although the density of the media has greatly increased, there has been little improvement in either access times or data transfer rates. In the case of optical storage, the access times have increased and the data transfer rates have decreased relative to magnetic media. Due to physical considerations, substantial increases in disk storage data transfer rates seem unlikely.

Because of increased processing power and the potential for high network transfer rates, new applications are emerging. These applications range from bulk data transfer for super computers to managing digital color video in real-time. Today, managing digitized color video in real-time is impossible. Storing just a few minutes of digitized color video requires gigabytes of storage. Storing or retrieving it in real-time requires sustained transfer rates on the order of 20 megabytes per second.

Our architecture, called Swift, addresses the problem of storing and retrieving large data objects from slow secondary storage at very high data-rates. Swift is based on the premises that: (1) the network interconnection will be capable of supporting much higher data-rates than individual storage agents; (2) resources can be preallocated for storing and transmitting data; (3) multiple storage agents can be driven concurrently using data striping; and (4) failures of storage agents can be masked using data redundancy.

Swift is based on a client-server model and addresses the issues of authentication, access control, and encryption. Since it is a distributed architecture made up of independently replaceable components, it can provide very high reliability. It is adaptable to different network interconnection topologies and technologies. Swift operates by having a storage mediator reserve resources from storage agents in a session-oriented manner, and then presenting a distribution agent with a *transfer plan*. The distribution agent stores or retrieves the data at the storage agents following that plan.

Even though Swift was designed with very large objects in mind, it can handle small objects such as those encountered in normal file systems with two penalties: one round trip time for a short network message to consult the storage mediator, and computing the required data redundancy. Swift is also well suited as a swapping device for high performance

Using Data Striping in a Local Area Network

Luis-Felipe Cabrera
IBM Almaden Research Center
Computer Science Department

Internet: cabrera@almaden.ibm.com

Darrell D. E. Long
Computer & Information Sciences
University of California at Santa Cruz

Internet: darrell@cis.ucsc.edu

Abstract

We use the technique of storing the data of a single object across several storage servers, called data striping, to achieve high transfer data rates in a local area network. Using parallel paths to data allows a client to transfer data to and from storage at a higher rate than that supported by a single storage server. We have implemented a network data service, called Swift, that uses data striping. Swift exhibits the expected scaling property in the number of storage servers connected to a network and in the number of interconnection networks present in the system.

We have also simulated a version of Swift to explore the limits of possible future configurations. We observe that the system can evolve to support very high speed interconnection networks as well as large numbers of storage servers. Since Swift is a distributed system made up of independently replaceable components, any component that limits the performance can either be *replaced* by a faster component when it becomes available or can be *replicated* and used in parallel. This should allow the system to incorporate and exploit emerging storage and networking technologies.

1 Introduction

The current generation of distributed computing systems do not support I/O-intensive applications well. In particular, they are incapable of integrating high-quality video with other data in a general purpose environment. For example, multimedia applications that require this level of service include scientific visualization, image processing, and recording and playback of color video. The data rates required by some of these applications range from 1.2

Swift/RAID: A Distributed RAID System

Darrell D. E. Long, Bruce R. Montague[†]
Computer and Information Sciences
University of California, Santa Cruz

Luis-Felipe Cabrera
Computer Science Department
IBM Almaden Research Center

Abstract

The Swift I/O architecture is designed to provide high data rates in support of multimedia type applications in general purpose distributed environments through the use of distributed striping. Striping techniques place sections of a single logical data space onto multiple physical devices. The original Swift prototype was designed to validate the architecture, but did not provide fault tolerance. We have implemented a new prototype of the Swift architecture that provides fault tolerance in the distributed environment in the same manner as RAID levels 4 and 5. RAID (Redundant Arrays of Inexpensive Disks) techniques have recently been widely used to increase both performance and fault tolerance of disk storage systems.

The new Swift/RAID implementation manages all communication using a distributed *transfer plan executor* which isolates all communication code from the rest of Swift. The transfer plan executor is implemented as a distributed finite state machine which decodes and executes a set of reliable data transfer operations. This approach enabled us to easily investigate alternative architectures and communications protocols.

Providing fault tolerance comes at a cost, since computing and administering parity data impacts Swift/RAID data rates. For a five node system, in one typical performance benchmark, Swift/RAID level 5 obtained 87% of the original Swift read throughput and 53% of the write throughput. Swift/RAID level 4 obtained 92% of the original Swift read throughput and 34% of the write throughput.

Keywords: Swift architecture, RAID, data striping, client-server data transmission, network data service, distributed atomic operations, concurrent programming, distributed state machines, real-time distributed programming.

1 Introduction

The Swift system was designed to investigate the use of network disk striping to achieve the data rates required by multimedia in a general purpose distributed system. The original Swift prototype was implemented during 1991, and its design and performance was described, investigated, and reported [Cabrera and Long, 1991, Emigh, 1992]. A high-level view of the Swift architecture is shown in Figure 1. Swift uses a high speed interconnection medium to aggregate arbitrarily many (slow) storage devices into a faster logical storage service, making all applications unaware of this aggregation. Swift uses a modular client-server architecture made up of independently replaceable components.

Disk striping is a technique analogous to main memory interleaving that has been used for some time to enhance throughput and balance disk load in disk arrays [Kim, 1986, Salem and Garcia-Molina, 1986]. In such systems writes scatter data across devices (the members of the stripe) while reads 'gather' data from

[†]Supported in part by the National Science Foundation under Grant NSF CCR-9111220 and by the Office of Naval Research under Grant N00014-92-J-1807

The Zebra Striped Network File System

by

John Henry Hartman

Sc. B. (Brown University) 1987
M.S. (University of California at Berkeley) 1990

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor John Ousterhout, Chair
Professor Randy Katz
Professor Ray Larson
Dr. Felipe Cabrera

1994

In *Proc. of the Extreme Linux Track: 4th Annual Linux Showcase and Conference*, October 2000.

PVFS: A Parallel File System for Linux Clusters*

Philip H. Carns Walter B. Ligon III
Parallel Architecture Research Laboratory
Clemson University, Clemson, SC 29634, USA
{pcarns, walt} @parl.clemson.edu

Robert B. Ross Rajeev Thakur
Mathematics and Computer Science Division
Argonne National Laboratory, Argonne, IL 60439, USA
{rross, thakur} @mcs.anl.gov

Abstract

As Linux clusters have matured as platforms for low-cost, high-performance parallel computing, software packages to provide many key services have emerged, especially in areas such as message passing and networking. One area devoid of support, however, has been parallel file systems, which are critical for high-performance I/O on such clusters. We have developed a parallel file system for Linux clusters, called the Parallel Virtual File System (PVFS). PVFS is intended both as a high-performance parallel file system that anyone can download and use and as a tool for pursuing further research in parallel I/O and parallel file systems for Linux clusters.

In this paper, we describe the design and implementation of PVFS and present performance results on the Chiba City cluster at Argonne. We provide performance results for a workload of concurrent reads and writes for various numbers of compute nodes, I/O nodes, and I/O request sizes. We also present performance results for MPI-IO on PVFS, both for a concurrent read/write workload and for the BTIO benchmark. We compare the I/O performance when using a Myrinet network versus a fast-ethernet network for I/O-related communication in PVFS. We obtained read and write bandwidths as high as 700 Mbytes/sec with Myrinet and 225 Mbytes/sec with fast ethernet.

*This work was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and in part by the National Aeronautics and Space Administration, under Research Grant NAG-5-3835.

1 Introduction

Cluster computing has recently emerged as a mainstream method for parallel computing in many application domains, with Linux leading the pack as the most popular operating system for clusters. As researchers continue to push the limits of the capabilities of clusters, new hardware and software have been developed to meet cluster computing's needs. In particular, hardware and software for message passing have matured a great deal since the early days of Linux cluster computing; indeed, in many cases, cluster networks rival the networks of commercial parallel machines. These advances have broadened the range of problems that can be effectively solved on clusters.

One area in which commercial parallel machines have always maintained great advantage, however, is that of parallel file systems. A production-quality high-performance parallel file system has not been available for Linux clusters, and without such a file system, Linux clusters cannot be used for large I/O-intensive parallel applications. We have developed a parallel file system for Linux clusters, called the Parallel Virtual File System (PVFS) [33], that can potentially fill this void. PVFS is being used at a number of sites, such as Argonne National Laboratory, NASA Goddard Space Flight Center, and Oak Ridge National Laboratory. Other researchers are also using PVFS in their studies [28].

We had two main objectives in developing PVFS. First, we needed a basic software platform for pursuing further research in parallel I/O and parallel file systems in the context of Linux clusters. For this purpose, we needed a stable, full-featured parallel file system to begin with. Our second objective was to meet the need for a paral-

This is G o o g l e's cache of http://hpcf.nersc.gov/training/tutorials/T3E/IO/striping.html.
G o o g l e's cache is the snapshot that we took of the page as we crawled the web.
The page may have changed since that time. Click here for the current page without highlighting.
To link to or bookmark this page, use the following url: http://www.google.com/search?q=cache:JKdv7JLCdGkC:hpcf.nersc.gov/training/tutorials/T3E/IO/striping.html+NERSC+Tutorials+Cray+T3E+disk&

Google is not affiliated with the authors of this page nor responsible for its content.

These search terms have been highlighted: **nersc tutorials cray t3e disk striping**

8. Disk striping

"Disk striping" refers to the process of storing a single file across multiple **disk** partitions. Each partition contains a different part of the file. **Disk striping** can greatly increase I/O since each partition can be accessed in parallel. **Striping** significantly improves I/O time for file sizes of 10s of MB or greater.

The `/usr/tmp` partition on *mcurie* now has system-level striping performed automatically across four disks. This removes much of the advantage of user-level striping as discussed below. The remaining text on this page was written before the installation of the current disks and is left here for reference.

You use the `assign` command to set parameters that control how the **striping** is performed. Fortran `OPEN()`, `READ()` and `WRITE()` statements are used as usual without modification. For illustration, we'll consider one PE writing a single file across multiple **disk** partitions on the **NERSC Cray T3E-900**.

Frangipani: A Scalable Distributed File System

Chandramohan A. Thekkath
Timothy Mann
Edward K. Lee

Systems Research Center
Digital Equipment Corporation
130 Lytton Ave, Palo Alto, CA 94301

Abstract

The ideal distributed file system would provide all its users with coherent, shared access to the same set of files, yet would be arbitrarily scalable to provide more storage space and higher performance to a growing user community. It would be highly available in spite of component failures. It would require minimal human administration, and administration would not become more complex as more components were added.

Frangipani is a new file system that approximates this ideal, yet was relatively easy to build because of its two-layer structure. The lower layer is Petal (described in an earlier paper), a distributed storage service that provides incrementally scalable, highly available, automatically managed virtual disks. In the upper layer, multiple machines run the same Frangipani file system code on top of a shared Petal virtual disk, using a distributed lock service to ensure coherence.

Frangipani is meant to run in a cluster of machines that are under a common administration and can communicate securely. Thus the machines trust one another and the shared virtual disk approach is practical. Of course, a Frangipani file system can be exported to untrusted machines using ordinary network file access protocols.

We have implemented Frangipani on a collection of Alphas running DIGITAL Unix 4.0. Initial measurements indicate that Frangipani has excellent single-server performance and scales well as servers are added.

1 Introduction

File system administration for a large, growing computer installation built with today's technology is a laborious task. To hold more files and serve more users, one must add more disks, attached to more machines. Each of these components requires human administration. Groups of files are often manually assigned to particular disks, then manually moved or replicated when components fill up, fail, or become performance hot spots. Joining multiple disk drives into one unit using RAID technology is only a partial solution; administration problems still arise once the system grows large enough to require multiple RAIDs and multiple server machines.

Frangipani is a new scalable distributed file system that manages a collection of disks on multiple machines as a single shared pool of storage. The machines are assumed to be under a common administration and to be able to communicate securely. There have been many earlier attempts at building distributed file systems that scale well in throughput and capacity [1, 11, 19, 20, 21, 22, 26, 31, 33, 34]. One distinguishing feature of Frangipani is that it has a very simple internal structure—a set of cooperating machines use a common store and synchronize access to that store with locks. This simple structure enables us to handle system recovery, reconfiguration, and load balancing with very little machinery. Another key aspect of Frangipani is that it combines a set of features that makes it easier to use and administer Frangipani than existing file systems we know of.

1. All users are given a consistent view of the same set of files.
2. More servers can easily be added to an existing Frangipani installation to increase its storage capacity and throughput, without changing the configuration of existing servers, or interrupting their operation. The servers can be viewed as "bricks" that can be stacked incrementally to build as large a file system as needed.
3. A system administrator can add new users without concern for which machines will manage their data or which disks will store it.
4. A system administrator can make a full and consistent backup of the entire file system without bringing it down. Backups can optionally be kept online, allowing users quick access to accidentally deleted files.
5. The file system tolerates and recovers from machine, network, and disk failures without operator intervention.

Frangipani is layered on top of Petal [24], an easy-to-administer distributed storage system that provides *virtual disks* to its clients. Like a physical disk, a Petal virtual disk provides storage that can be read or written in blocks. Unlike a physical disk, a virtual disk provides a sparse 2^{64} byte address space, with physical storage allocated only on demand. Petal optionally replicates data for high availability. Petal also provides efficient snapshots [7, 10] to support consistent backup. Frangipani inherits much of its scalability, fault tolerance, and easy administration from the underlying storage system, but careful design was required to extend these

Kai Hwang and Hai Jin
University of Southern California

Edward Chow and Cho-Li Wang
University of Hong Kong

Zhiwei Xu
Chinese Academy of Sciences

Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space

Adopting a new hierarchical checkpointing architecture, the authors develop a single I/O address space for building highly available clusters of computers. They propose a systematic approach to achieving single system image by integrating existing middleware support with the newly developed features.

The computing trend is moving from clustering high-end mainframes to clustering desktop computers. This trend is triggered by the widespread use of PCs, workstations, gigabit networks, and middleware support for clustering.[1] This article presents new approaches to achieving fault tolerance and *single system image* (SSI) in a workstation cluster. In a cluster of computers, local area networks or high-bandwidth switch networks using optical fibers physically connect a collection of node computers. The workstations in a cluster can work collectively as an integrated computing resource—that is, an SSI—or they can operate as individual computers, separately.

Present clusters are usually small and provide only limited SSI services. Future clusters will likely increase in scalability and offer more SSI support, as Figure 1 illustrates. The implication is that future clusters could replace the MPP, SMP, or CC-NUMA architectures (see "The cluster as a computer architecture" sidebar for key characteristics of these computer platforms).

We focus on clusters with high availability through SSI support, distributed RAID (redundant arrays of inexpensive disks) with parity checks, and hierarchical checkpointing with adaptive recovery. In particular, we developed a single I/O address space among all disks and peripheral devices attached in the cluster. This enables direct remote disk access, which is a necessary step to implement a

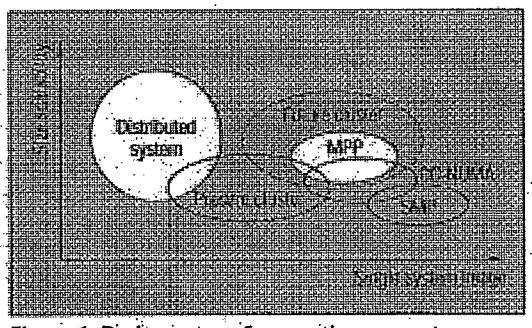


Figure 1. Design space of competing computer architectures.

1092-3063/99/$10.00 © 1999 IEEE

Introducing Microsoft Cluster Service (MSCS) in the Windows Server 2003 Family

Page Options
Average rating:
6 out of 9
Rate this page
Print this page
E-mail this page
Add to Favorites

Mohan Rao Cavale
Microsoft Corporation

November 2002

Applies to:
- Windows® Server 2003
- Windows® Server 2003, Enterprise Edition
- Windows® Server 2003, Datacenter Edition
- Microsoft Cluster Service

Summary: Learn how to easily perform a sanity check of your application within a cluster environment without having to make any changes to your application's code. This paper focuses on Cluster Service, one of three Microsoft server technologies that support clustering. (15 printed pages)

Contents

Introduction

Delivering a great quality application with a rich feature set isn't enough in all cases—increasingly, it must also meet high availability criteria. Have you avoided taking your application to the next level because cluster technology seems too daunting to understand and use? With Microsoft's® Cluster Service— introduced in Windows® NT™ 4 and available in the Windows Server 2003 family, developers have at their disposal straightforward tools to deploy applications in a clustered environment. These include the ability to enlist any application in a cluster as a generic application, and the ability to control application configuration by means of Window scripting.

A cluster connects two or more servers together so that they appear as a single computer to clients. Connecting servers in a cluster allows for workload sharing, enables a single point of operation/management, and provides a path for scaling to meet increased demand. Thus, clustering gives you the ability to produce high availability applications.

This paper focuses on Cluster Service, one of three Microsoft server technologies that support clustering. We demonstrate how to easily perform a sanity check of your application within a cluster environment without having to make any changes to your application's code.

Three Technologies for Clustering

Microsoft servers provide three technologies to support clustering: Network Load Balancing (NLB), Component Load Balancing (CLB), and Microsoft Cluster Service (MSCS).

Network Load Balancing

Network Load Balancing acts as a front-end cluster, distributing incoming IP traffic across a cluster of servers, and is ideal for enabling incremental scalability and outstanding availability for e-commerce Web sites. Up to 32 computers running a member of the Windows Server 2003 family can be connected to share a single virtual IP address. NLB enhances scalability by distributing its client requests across multiple servers within the cluster. As traffic increases, additional servers can be added to the cluster; up to 32 servers are possible in any one cluster. NLB also provides high availability by automatically detecting the failure of a server and repartitioning client traffic among the remaining servers within 10 seconds, while it provides users with continuous service.

Edgar H. Sibley
Panel Editor

Using only a few simple and commonplace instructions, this algorithm efficiently maps variable-length text strings onto small integers.

Fast Hashing of Variable-Length Text Strings

Peter K. Pearson

In the literature on hashing techniques, most authors spend little time discussing any particular hashing function, but make do with an allusion to Knuth [3] in their haste to get to the interesting topics of table organization and collision resolution. The relatively rare articles on hashing functions themselves [2] tend to discuss algorithms that operate on values of predetermined length or that make heavy use of operations (multiplication, division, or shifts of long bit strings) that are absent from the instruction sets of smaller microprocessors.

This article proposes a hashing function specifically tailored to variable-length text strings. This function takes as input a word W consisting of some number n of characters, $C_1, C_2, \ldots, C_n$, each character being represented by one byte, and returns an index in the range 0–255. An auxiliary table T of 256 randomish bytes is used in the process. Here is the proposed algorithm:[1]

```
h[0] := 0 ;
for i in 1..n loop
   h[i] := T[ h[i-1] xor C[i] ] ;
end loop ;
return h[n] ;
```

Notice that the processing of each additional character of text requires only an exclusive-OR operation and an indexed memory read. Also note that it is not necessary to know the length of the string at the beginning of the computation, a property useful when the end of the text string is indicated by a special character rather than by a separately stored length variable.

Two desirable properties of this algorithm for hashing variable-length strings derive from the technique of *cryptographic checksums* or *message authentication codes* [4], from which it is adapted. First, a good cryptographic checksum ensures that small changes to the data result in large and seemingly random changes to the checksum. In the hashing adaptation, this results in good separation of very similar strings. Second, on a good cryptographic checksum the effect of changing one part of the data must *not* be cancelled by an easily computed change to some other part. In hashing, this ensures good separation of anagrams, the downfall of hashing strategies that begin with a length-reducing exclusive-OR of substrings.

The auxiliary table T is obviously crucial to this algorithm, yet I have found very few constraints on its construction. Since the hashing function can only return values that appear in T, each index from 0 to 255 must appear in T exactly once. In other words, T must be a permutation of the values (0 . . . 255). Obviously, if T[i] = i, the corresponding h is merely a longitudinal exclusive-OR checksum, which is a bad hashing function because it does not separate anagrams. I have experimented by filling T with randomly generated permutations of (0 . . . 255) and have found no outstanding good or bad arrangements. (An attempt to promote greater dispersal among very similar short strings by clever choice of T, however, turned out to be a very bad idea.)

For the interested reader who does not want to generate his own random permutations, Table I presents the permutation used in the tests described later in this article.

SEPARATION PERFORMANCE

The purpose of any text hashing function is to take text strings—even very similar text strings—and map them onto integers that are spread as uniformly as possible over the intended range of output values. In the absence of prior knowledge about the strings being hashed, a perfectly uniform output distribution cannot be expected. The best result that one can expect to achieve consistently is a seemingly random mapping of input strings onto output values. To see how well h does its job, one might ask the following questions.

- If h is applied to a string of random bytes, is each of the 256 possible outcomes equally likely? The answer, probably not surprisingly, is yes. From the algorithm given earlier, it is clear that if the last input character, C[n], is *random*—equally likely to take any value, and uncorrelated with any preceding character—then all final values of h are equally likely.
- If two input strings differ by a single bit, will their hash function values collide more often than by

[1] In a practical implementation, the subscripts on h are omitted. They are shown here to clarify later discussion.

© 1990 ACM 0001-0782/90/0600-0677 $1.50

Installation and Administration

Kimberlite Cluster Version 1.1.0

Revision D

Copyright © 2000 K. M. Sorenson

December, 2000

This document describes how to set up and manage a Kimberlite cluster, which provides application availability and data integrity. Send comments to documentation@missioncriticallinux.com.

The HP AutoRAID hierarchical storage system

John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan
Hewlett-Packard Laboratories

Configuring redundant disk arrays is a black art. To configure an array properly, a system administrator must understand the details of both the array and the workload it will support. Incorrect understanding of either, or changes in the workload over time, can lead to poor performance.

We present a solution to this problem: a two-level storage hierarchy implemented inside a single disk-array controller. In the upper level of this hierarchy, two copies of active data are stored to provide full redundancy and excellent performance. In the lower level, RAID 5 parity protection is used to provide excellent storage cost for inactive data, at somewhat lower performance.

The technology we describe in this paper, known as HP AutoRAID, automatically and transparently manages migration of data blocks between these two levels as access patterns change. The result is a fully redundant storage system that is extremely easy to use, is suitable for a wide variety of workloads, is largely insensitive to dynamic workload changes, and performs much better than disk arrays with comparable numbers of spindles and much larger amounts of front-end RAM cache. Because the implementation of the HP AutoRAID technology is almost entirely in software, the additional hardware cost for these benefits is very small.

We describe the HP AutoRAID technology in detail, provide performance data for an embodiment of it in a storage array, and summarize the results of simulation studies used to choose algorithms implemented in the array.

Categories and Subject Descriptors: B.4.2 [**Input/Output and Data Communications**]: Input/Output devices—*channels and controllers*; B.4.5 [**Input/Output and Data Communications**]: Reliability, Testing, and Fault-Tolerance—*redundant design*; D.4.2 [**Operating Systems**]: Storage Management—*secondary storage*

General Terms: Algorithms, Design, Performance, Reliability

Additional Key Words and Phrases: Disk array, RAID, storage hierarchy

1. INTRODUCTION

Modern businesses and an increasing number of individuals depend on the information stored in the computer systems they use. Even though modern disk drives have mean-time-to-failure (MTTF) values measured in hundreds of

Author's addresses: Hewlett-Packard Laboratories, mailstop 1U13, 1501 Page Mill Road, Palo Alto, CA 94304-1126; email: {wilkes,golding,staelin,sullivan}@hpl.hp.com.
Permission to make digital/hard copy of all or part of this material without fee is granted provided that the copies are not made or distributed for profit or commercial advantage; the ACM copyright/server notice, the title of the publication, and its date appear; and notice is given that copying is by permission of the Association for Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 1996 ACM 0734-2071/96/0200-0108 $03.50

AFRAID — A Frequently Redundant Array of Independent Disks

Stefan Savage
University of Washington, Seattle, WA

John Wilkes
Hewlett-Packard Laboratories, Palo Alto, CA

Abstract

Disk arrays are commonly designed to ensure that stored data will *always* be able to withstand a disk failure, but meeting this goal comes at a significant cost in performance. We show that this is unnecessary. By trading away a fraction of the enormous reliability provided by disk arrays, it is possible to achieve performance that is almost as good as a non-parity-protected set of disks.

In particular, our AFRAID design eliminates the small-update penalty that plagues traditional RAID 5 disk arrays. It does this by applying the data update immediately, but delaying the parity update to the next quiet period between bursts of client activity. That is, AFRAID makes sure that the array is *frequently* redundant, even if it isn't always so. By regulating the parity update policy, AFRAID allows a smooth trade-off between performance and availability.

Under real-life workloads, the AFRAID design can provide close to the full performance of an array of unprotected disks, and data availability comparable to a traditional RAID 5. Our results show that AFRAID offers 42% better performance for only 10% less availability, 97% better for 23% less, and as much as a factor of 4.1 times better performance for giving up less than half RAID 5's availability.

We explore here the detailed availability and performance implications of the AFRAID approach.

1. Introduction

In a RAID 5 disk array, small writes take a long time to complete [Patterson88]. This is known as the "small update problem". In such an array, redundancy for a stripe of data is provided by a parity block, computed as the XOR of the data blocks in the stripe, in order to allow recovery if any disk fails. If a portion of a stripe is updated, the parity data must also be updated to preserve the recoverability property (Figure 1). To do this, it is necessary to (1) read the old value of the data to be overwritten, unless it is already cached in the array controller; (2) read the old parity; (3) XOR the new data with the old, and XOR the result with the old parity to generate the new parity data; (4) write the new data and (5) write the new parity.

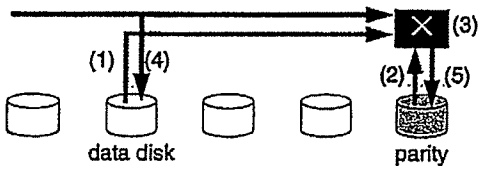


Figure 1: doing a small update in a traditional RAID 5.

Thus, three or four disk I/Os are needed to achieve one small write — all of which are in the critical path. In contemplating this problem we made the following observations:

- modern disks are extremely reliable—so much so that disk array reliability is limited more by its support components than its disks;
- many real workloads have slack periods between bursts of client activity;
- people are already well-used to the notion of time-limited exposure to risk.

These eventually led us to the idea of AFRAID (*A Frequently Redundant Array of Independent Disks*).[1] AFRAID is a RAID 5 disk array that relaxes the coherency between data and parity for short periods of time; parity is made consistent again in the idle periods between bursts of client writes. Thus the stored data is *frequently* held redundantly, rather than always guaranteed to be so.

In this approach, small updates are not required to wait for the parity to be updated, thereby reducing the four I/Os in the critical path of the traditional small-update protocol to just one: write the new data. The benefit is that performance approaches that of an unprotected array. The disadvantage is a slightly increased risk of data loss from a disk failure, but we will show that this increase is small in practice, and also that it can be bounded at the cost of some performance. That is, AFRAID allows a smooth trade-off between increased reliability and increased performance.

[1] Like so many good ideas, ours was of course developed by back-determination from the acronym.

JTK:klw

Please Date Stamp and Return

The Commissioner for Patents has received from Bromberg & Sunstein LLP the following re:

Inventor:	Miloushev, Vladimir	Docket No.:	3193/102
Title:	File Switch and Switched File System	Art Unit:	2142
		Examiner	Prieto, Beatriz
Serial/Patent No.:	10/043,413	Date:	October 30, 2007
Filing/Issue Date:	January 10, 2002	Express Mail No.:	

Documents:

- ☐ New Application Transmittal
- ☐ Provisional Application Cover Sheet
- ☐ Description- pages.
- ☐ Claims- pages
- ☐ Abstract
- ☐ Application Data Sheet
- ☐ Request and Certification under 35 USC 122(b)(2)(B)(i)
- ☐ sheets of drawings
  - ☐ formal ☐ informal
- ☐ Declaration & Power of Attorney
  - ☐ executed ☐ unexecuted
- ☐
- ☐

- ☐ Amendment Transmittal
- ☐ Amendment (Preliminary)
- ☒ Request for Continued Examination
- ☒ IDS & References
- ☐ Petition for month extension
- ☐ Issue Fee Transmittal & Form PTOL-85b
- ☐ Payment of Maintenance Fee
- ☐ Assignment/Recordation Form Cover Sheet
- ☐ Check in the amount of $
- ☐ Completion of Filing Requirements
- ☐ Transmittal of Formal Drawings
- ☒ $405 00 charge to deposit account
- ☐